

VEISPH code: two dimensional incompressible SPH code for viscoelastic flows

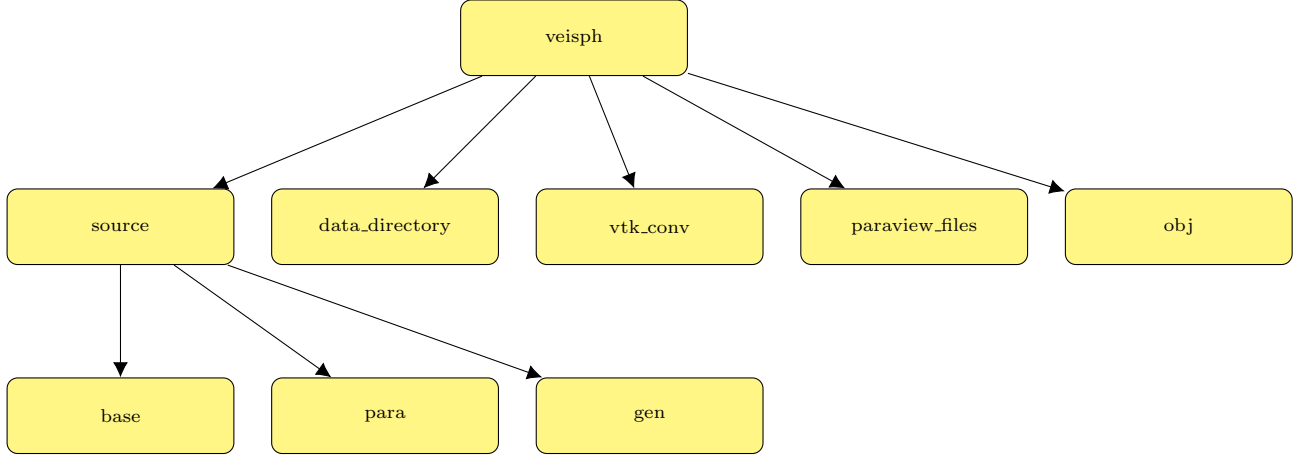
J. R. C. King*

*Department of Mechanical, Aerospace and Civil Engineering,
The University of Manchester, Manchester, M13 9PL, UK*

(Dated: February 12, 2021)

A brief description of the VEISPH code. Numerical methods in this code follow arXiv:2009.12245 [1]

I. FILE STRUCTURE



- **source** contains source files for the code:
 - **base** contains main modules for VEISPH
 - **para** contains parameters and common variables
 - **gen** contains code to generate casefiles and initial conditions
- **data_directory** contains output files produced by the code
- **vtk_conv** contains a program to convert output files into **.vtu** files, which can be read by Paraview.
- **paraview_files** **.vtu** files are automatically created in here.
- **obj** contains **.o** and **.mod** files created during compilation.

II. GOVERNING EQUATIONS

The governing equations (here listed in dimensionless form) in the arbitrary frame of reference as in [1] are:

$$\nabla \cdot \mathbf{u} = 0 \quad (1a)$$

$$\frac{d\mathbf{u}}{dt} - \mathbf{u}_s \cdot \nabla \cdot \mathbf{u} = -\nabla p + \beta \sqrt{\frac{Pr}{Ra}} \nabla^2 \mathbf{u} + \frac{(1-\beta)Pr}{Ra \times El} \nabla \cdot \boldsymbol{\tau}_p + \theta \mathbf{e}_y + \mathbf{f} \quad (1b)$$

$$\frac{d\mathbf{A}}{dt} - \mathbf{u}_s \cdot \nabla \mathbf{A} - (\mathbf{A} \cdot \nabla \mathbf{u}^T + \nabla \mathbf{u} \cdot \mathbf{A}) = \frac{-1}{El} \sqrt{\frac{Pr}{Ra}} f_R(\mathbf{A}) \quad (1c)$$

$$\frac{d\theta}{dt} - \mathbf{u}_s \cdot \nabla \theta = \frac{1}{\sqrt{Ra \times Pr}} \nabla^2 \theta \quad (1d)$$

* jack.king@manchester.ac.uk

TABLE I. Strain and relaxation functions for the constitutive models used in this work.

Constitutive model	f_S	f_R
Oldroyd B	$\mathbf{A} - \mathbf{I}$	$\mathbf{A} - \mathbf{I}$
FENE-P	$\frac{\mathbf{A}}{1 - \text{tr}(\mathbf{A})/L^2} - \mathbf{I}$	$\frac{\mathbf{A}}{1 - \text{tr}(\mathbf{A})/L^2} - \mathbf{I}$
FENE-CR	$\frac{\mathbf{A} - \mathbf{I}}{1 - \text{tr}(\mathbf{A})/L^2}$	$\frac{\mathbf{A} - \mathbf{I}}{1 - \text{tr}(\mathbf{A})/L^2}$
Linear PTT	$\mathbf{A} - \mathbf{I}$	$[1 + \varepsilon \text{tr}(\mathbf{A} - \mathbf{I})](\mathbf{A} - \mathbf{I})$
Exponential PTT	$\mathbf{A} - \mathbf{I}$	$\exp[\varepsilon \text{tr}(\mathbf{A} - \mathbf{I})](\mathbf{A} - \mathbf{I})$
Giesekus	$\mathbf{A} - \mathbf{I}$	$\alpha \mathbf{A}^2 + (1 - 2\alpha)\mathbf{A} - (1 - \alpha)\mathbf{I}$

where \mathbf{u} is the velocity, p the pressure, θ the temperature deviation (from ambient), and $\boldsymbol{\tau}_p$ is the polymeric stress, related to the conformation tensor \mathbf{A} by the strain function $\boldsymbol{\tau}_p = f_S(\mathbf{A})$. f_R is a relaxation function. \mathbf{f} is a body force.

N.B. Equation (1d) is not yet implemented in the code!

The system is controlled by the four dimensionless quantities:

$$\beta = \eta_s / \eta_0 \quad \text{Viscosity ratio} \quad (2a)$$

$$Pr = \frac{c_p \eta_0}{\kappa} \quad \text{Prandtl number} \quad (2b)$$

$$Ra = \frac{L^3 \Delta \rho |\mathbf{g}|}{\alpha \eta_0} \quad \text{Rayleigh number} \quad (2c)$$

$$El = \frac{\lambda \eta_0}{L^2} \quad \text{Elasticity number,} \quad (2d)$$

where η_s is the solvent viscosity, η_0 is the total viscosity, c_p is the specific heat capacity (at const. pressure), κ is the thermal diffusivity, α is the thermal conductivity (related to κ via density ρ and c_p), \mathbf{g} is the acceleration due to gravity, $\Delta \rho$ is a characteristic density deviation, λ is the relaxation time and L is a characteristic length-scale.

The Reynolds number is related to these dimensionless groups by: $Re = \sqrt{Pr/Ra}$. The Weissenberg number is $Wi = El\sqrt{Ra/Pr}$.

The strain and relaxation functions for various constitutive models are given in Table I.

Time evolution is with a first-order projection scheme [2], with divergence free velocity constraint enforced via a Poisson equation, which is solved using a BiCGStab algorithm with Jacobi preconditioner. Boundary conditions are imposed with mirror particles. SPH gradient and divergence operators are corrected to first order following [3]. Temporal evolution of the conformation tensor is via the log-conformation formulation of [4, 5], mostly following [6].

III. ELASTO-VISCOUS STRESS SPLITTING

With the above non-dimensionalisation, we introduce the tensor

$$\boldsymbol{\Phi} = \boldsymbol{\tau}_p - \frac{\alpha_{evss} El}{1 - \beta} \sqrt{\frac{Ra}{Pr}} (\nabla \mathbf{u} + \nabla \mathbf{u}^T) \quad (3)$$

and then express (1b) as

$$\frac{d\mathbf{u}}{dt} - \mathbf{u}_s \cdot \nabla \cdot \mathbf{u} = -\nabla p + (\beta + \alpha_{evss}) \sqrt{\frac{Pr}{Ra}} \nabla^2 \mathbf{u} + \frac{(1 - \beta) Pr}{Ra \times El} \nabla \cdot \boldsymbol{\Phi} + \theta \mathbf{e}_y + \mathbf{f}. \quad (4)$$

This formulation introduces a small amount of additional viscosity into the solvent part, and then removes the same amount via the divergence of $\boldsymbol{\Phi}$, improving stability when β is small or zero. For $\beta > 0.1$ we can usually set $\alpha_{evss} = 0$, and (4) returns to (1b). For smaller β , values of α_{evss} between 0.01 and 0.1 are usually sufficient to provide stability.

IV. CASE CREATION

In the directory `source/gen/` there is a file `datclass.F90`. This is the main part of a program which generates the input data for the code. Navigate to this directory, and then build and run it with:

```
make
./gen2D
cp I* ../././.
```

When you run `./gen2D`, it will give a list of cases and prompt you to choose one. Type the relevant number and press enter. You can modify `datclass.F90` to create more cases.

The following cases are currently included in `datclass.F90`:

1. 2D dam break
2. Taylor Green vortices
3. Poiseuille flow
4. Taylor-Couette flow
5. Periodic cylinders in a channel (as in [7])
6. Kolmogorov flow (you need to set the `external_forcing` switch to `.true.`).
7. Plane Couette flow

V. COMPILING AND RUNNING THE CODE

The Makefile is build for systems with the compiler `gfortran` install. If using an alternative compiler, modify the lines `FC := gfortran` and `LD := gfortran` to point to your chosen compiler. You need a system with OpenMP installed.

To compile the code, use `make const_mod=X` where `X` points to the constitutive model of you want. 1 gives Newtonian, 2 to 7 give Oldroyd B, FENE-P, FENE-CR, Linear PTT, Exponential PTT and Giesekus (respectively). Oldroyd B is the simplest viscoelastic constitutive model, and first to experiment with.

To run the code (after having created the case files as described in Section ??), run `./veisph`

VI. BOUNDARY FRAMEWORK

The computational domain is described by a set of boundary nodes, connected by straight lines (boundary patches), along with circles. In `datclass.F90` these are hard-coded for each case. For example, a rectangular domain with solid upper and lower boundaries and periodic lateral boundaries would be defined as:

```
b_type(:) = (/ 1, 2, 1, 2/)
b_vel(:) = (/ -0.0d0, 0.0d0, 0.0d0, 0.0d0/)
b_periodic_parent(:) = (/ 3, 4, 1, 2/)
b_node(1,:) = (/ 0.0d0, 0.0d0 /)
b_node(2,:) = (/ x1, 0.0d0 /)
b_node(3,:) = (/ x1, y1 /)
b_node(4,:) = (/ 0.0d0, y1 /)
```

where `x1` and `y1` are variables describing the length and height of the domain. The array `b_vel` indicates the velocity (tangential to the boundary patch) and here is usually zero. The array `b_type` indicates whether the patch is a wall (1) periodic (2), invisible (0) or (in other versions, but not relevant to this project) inflow (3) or outflow (4). For periodic patches, a relationship needs to be defined with a parent patch, which is done via the array `b_periodic_parent`.

Circular obstacles are described similarly, with centre `c_centre`, radius `c_radius`, angular velocity `c_omega` and translational velocity `c_vel`. In this project, the translational velocities will probably always be zero. A positive value of `c_radius` indicates the circle is an internal obstacle (e.g. the inner boundary in Taylor-Couette flow), whilst a negative value indicates the circle is an external boundary (e.g. the outer boundary in Taylor-Couette flow).

A. How boundary conditions are applied in the code

In the code (`source/base/mirror_boundaries_mod.F90`), the routine runs through all particles and identifies those near boundaries. Then, it runs through each boundary, and for all particles near that boundary, creates a mirror particle in the appropriate place, with appropriate conditions. The code then runs through all corners (i.e. all boundary nodes), and performs a similar procedure. Each mirror particle j has a parent particle i , and the array `irelation` tracks this: `irelation(j)=i`. The array `vrelation` stores information (in a confusing way) about the type of boundary which relates a mirror and its parent, and is used in specifying velocity relationships.

For example, if particle i is near a solid boundary patch, a particle j will be created which is a reflection of particle i in the boundary patch. Particle j will have velocity $\mathbf{u}_j = 2\mathbf{u}_b - \mathbf{u}_i$, where \mathbf{u}_b is the velocity of the boundary patch.

Pressure boundary conditions ($\mathbf{n} \cdot \nabla p = \mathbf{f} \cdot \mathbf{n}$) are constructed by specifying the difference between a mirror and its parent pressure through the array `dp_mp`, such that $P(i) = P(j) + dp_mp(i)$.

VII. OVERVIEW OF THE CODE STRUCTURE

The code consists of modules, each module is stored in a separate `.F90` file. Each module contains one or more subroutines which perform similar tasks, or tasks related to a theme (e.g. the module `part_shift` contains routines related to Fickian shifting).

The main program is contained within `sph2D_incom.F90`. It calls routines from the input module `input.F90`, some other housekeeping, then contains the main time loop. Within the main time loop, the routine `div_free` is called, which performs a single time step, then `output` is called, which saves any data as required. The module `div_free.F90` is the heart of the code, and is fairly clearly commented, the the subroutine `divfree` corresponds (roughly) to the steps 1 to 7 on pages 5 and 6 of [1].

VIII. OUTPUTS

Data from the code are saved in the folder `data_directory`. Within the folder `vtk_conv` there is a small program which converts the data into a format which can be read by Paraview.

-
- [1] J. King, S. Lind, High Weissenberg number simulations with incompressible Smoothed Particle Hydrodynamics and the log-conformation formulation, 2020. [arXiv:2009.12245](https://arxiv.org/abs/2009.12245).
 - [2] A. J. Chorin, Numerical solution of the Navier Stokes equations, *Journal of Mathematical Computing* 22 (1968) 745–762.
 - [3] J. Bonet, T.-S. Lok, Variational and momentum preservation aspects of Smooth Particle Hydrodynamic formulations, *Computer Methods in Applied Mechanics and Engineering* 180 (1999) 97 – 115. URL: <http://www.sciencedirect.com/science/article/pii/S0045782599000511>. doi:doi:[https://doi.org/10.1016/S0045-7825\(99\)00051-1](https://doi.org/10.1016/S0045-7825(99)00051-1).
 - [4] R. Fattal, R. Kupferman, Constitutive laws for the matrix-logarithm of the conformation tensor, *Journal of Non-Newtonian Fluid Mechanics* 123 (2004) 281 – 285. doi:doi:[10.1016/j.jnnfm.2004.08.008](https://doi.org/10.1016/j.jnnfm.2004.08.008).
 - [5] R. Fattal, R. Kupferman, Time-dependent simulation of viscoelastic flows at high Weissenberg number using the log-conformation representation, *Journal of Non-Newtonian Fluid Mechanics* 126 (2005) 23 – 37. doi:doi:[10.1016/j.jnnfm.2004.12.003](https://doi.org/10.1016/j.jnnfm.2004.12.003).
 - [6] J. López-Herrera, S. Popinet, A. Castrejón-Pita, An adaptive solver for viscoelastic incompressible two-phase problems applied to the study of the splashing of weakly viscoelastic droplets, *Journal of Non-Newtonian Fluid Mechanics* 264 (2019) 144 – 158. doi:doi:<https://doi.org/10.1016/j.jnnfm.2018.10.012>.
 - [7] A. Vázquez-Quesada, M. Ellero, SPH simulations of a viscoelastic flow around a periodic array of cylinders confined in a channel, *Journal of Non-Newtonian Fluid Mechanics* 167-168 (2012) 1 – 8. doi:doi:<https://doi.org/10.1016/j.jnnfm.2011.09.002>.